Lab #2
Data visualization

In this lab, we are going to work with a very popular time course cDNA data set from Paul Spellman's lab at Stanford. This microarray was designed with probes from the yeast *Saccharomyces cerevisiae* genome. The data set includes 3 different experiments, each with its own time course (each array is a different time point) for measuring transcript levels that are induced by various cyclins. The transcripts that respond to this stimulus are seen to be regulated at the different stages of the cell cycle. The 3 experiments differ by the method that the yeast cultures were synchronized: α factor arrest, elutriation, and arrest of a *cdc15* temperature-sensitive mutant. The *cdc15* time course experiment is the one that we will use in this lab to conduct some simple mathematical manipulations and plots.

The paper, "Comprehensive Identification of Cell Cycle–regulated Genes of the Yeast *Saccharomyces cerevisiae* by Microarray Hybridization" is available on the course website.

1.) Go to class website under Course Documents > Data Sets and download the Spellman yeast cell cycle dataset zip file.

2.) Unzip the text file, and read into R (Hint: using the read.table() function with a "header=T" argument and "row.names=1" argument is one method to do this).
> lab2=read.table("/Users/stevendea/Desktop/JHU/Fall 2019/Gene Expression Data Analysis and Visualization/Labs/Lab2/spellman.txt", header = TRUE, row.names=1)

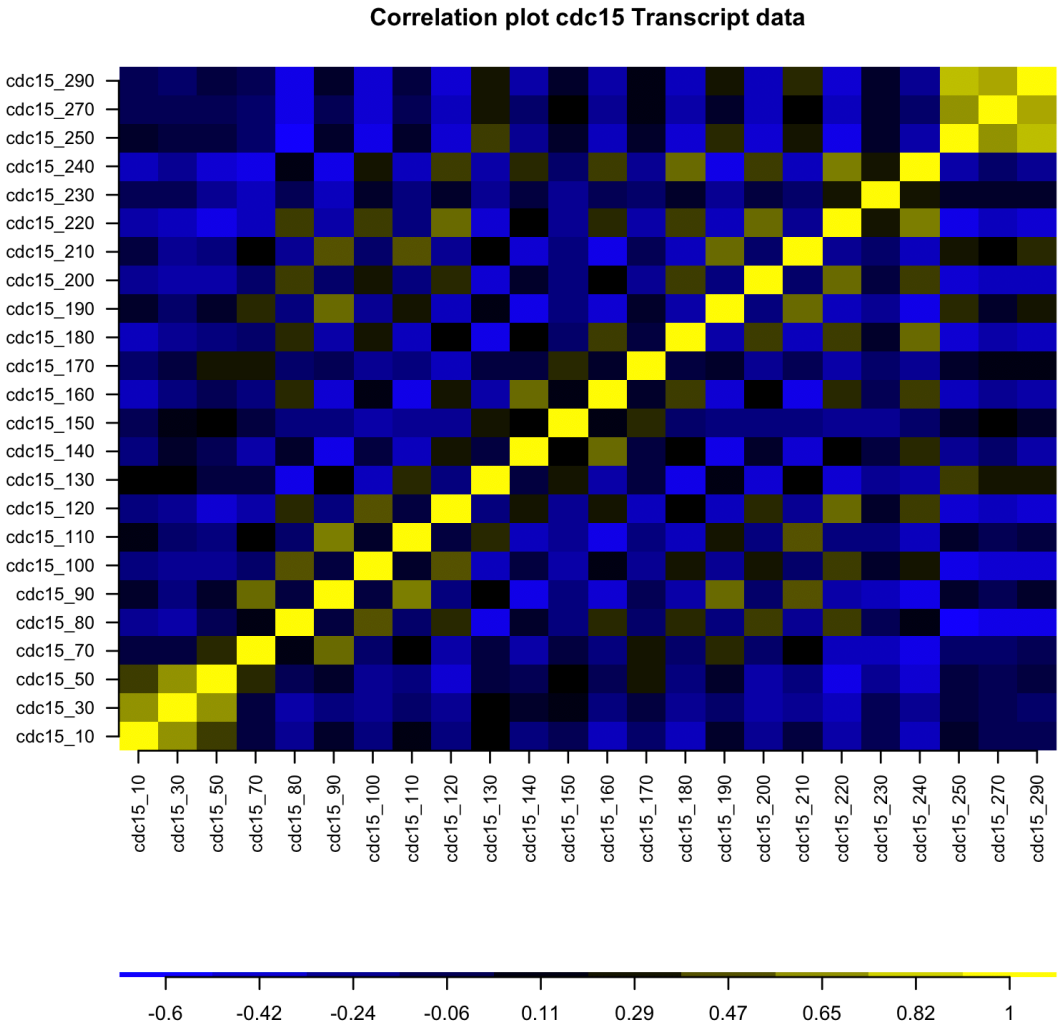
3.) Look at the dimensions of the data frame and make sure that there are 6,178 genes and 77 arrays/sample.
> dim(lab2)
[1] 6178 77

4.) Isolate only the cdc15 experiment (samples 23-46).
> cdc15=lab2[23:46]

5). Now calculate a correlation matrix between the time points (use Pearson's correlation). Make sure to title the plot, label the axes, and provide a legend of the color gradient. In the correlation calculation, make sure to use the argument 'use' and value=pairwise.complete.obs since all of these arrays have at least one missing value.

```
> cdc15.cor <- cor(cdc15, use="pairwise.complete.obs")
> layout(matrix(c(1,1,1,1,1,1,1,1,2,2), 5, 2, byrow = TRUE))
> par(oma=c(5,7,1,1))
> cx <- rev(colorpanel(25,"yellow","black","blue"))
> leg <- seq(min(cdc15.cor,na.rm=T),max(cdc15.cor,na.rm=T),length=10)
> image(cdc15.cor,main="Correlation plot cdc15 Transcript data",axes=F,col=cx)
```

```
>
axis(1,at=seq(0,1,length=ncol(cdc15.cor)),label=dimnames(cdc15.cor)[[2]],cex.axis=0
.9,las=2)
>
axis(2,at=seq(0,1,length=ncol(cdc15.cor)),label=dimnames(cdc15.cor)[[2]],cex.axis=0
.9,las=2)
> image(as.matrix(leg),col=cx,axes=F)
> tmp <- round(leg,2)
> axis(1,at=seq(0,1,length=length(leg)),labels=tmp,cex.axis=1)
```

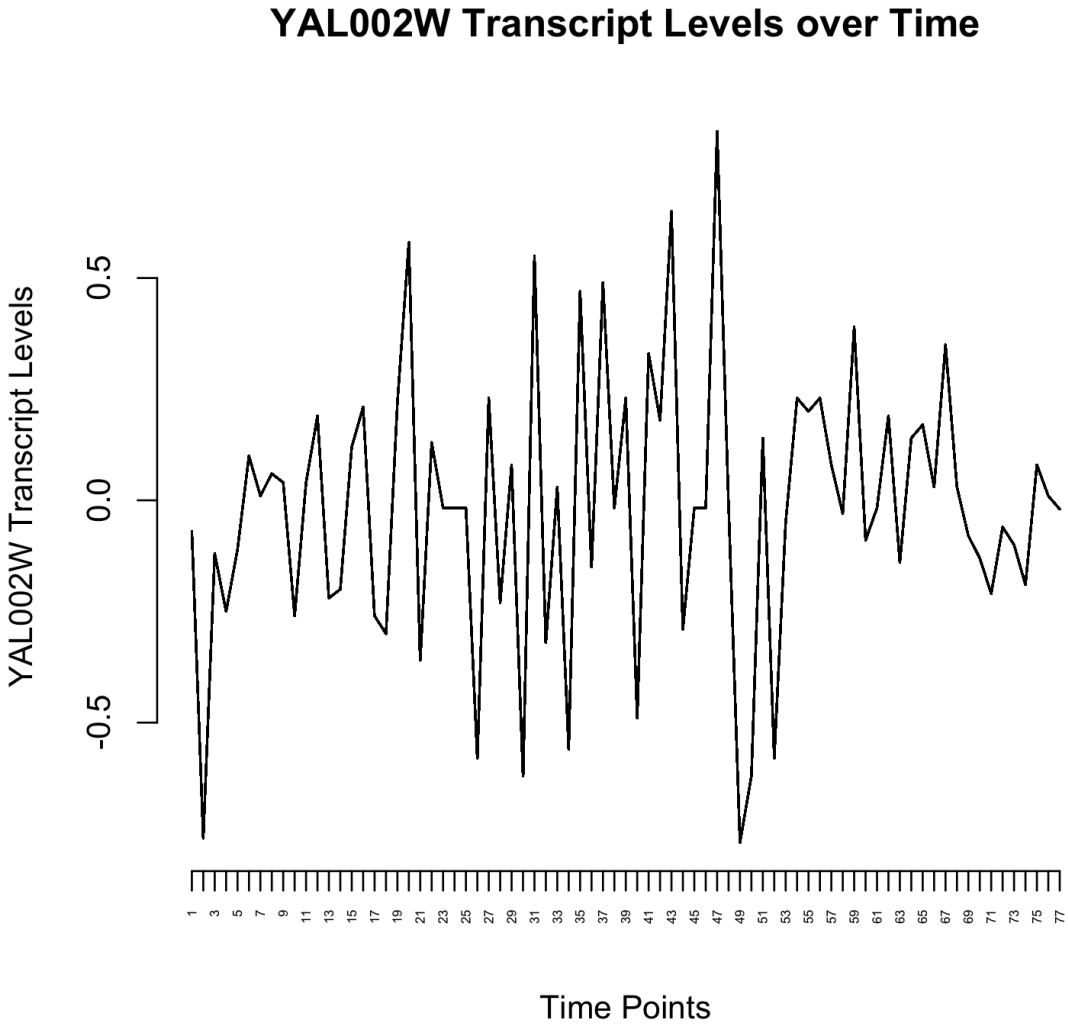


6.) Select the gene YAL002W, which is VPS8, a membrane-binding component of the CORVET complex; involved in endosomal vesicle tethering and fusion in the endosome to vacuole protein targeting pathway. Impute the missing values with the row mean (save as something). Make sure to cast the gene to numeric.

```
> YAL002W <- lab2["YAL002W",]
> mean(YAL002W.n, na.rm=TRUE)
[1] -0.017
> YAL002W.nmean <- mean(YAL002W.n, na.rm=TRUE)
> YAL002W.n[is.na(YAL002W.n)] <- YAL002W.nmean
```

7.) Generate a profile plot of the same gene. Title the plot, label the axes, and on the x-axis, provide the time points only for each array (no "cdc15_" prefix) so we can visualize the transcript pattern over time. Use lwd in the plot command (lwd=line width).

```
> plot(YAL002W.n, type='n', main ="YAL002W Transcript Levels over Time",
xlab= "Time Points", ylab="YAL002W Transcript Levels",axes=F, lwd=1)
> axis(side=1, at=c(1:77), labels=dimnames(YAL002W.n), cex.axis=0.4, las=2)
> axis(side=2)
> for(i in 1:length(YAL002W.n)){lines(YAL002W.n)}
```


YAL002W Transcript Levels over Time
YAL002W Transcript Levels
0.5
0.0
-0.5
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
35
37
39
41
43
45
47
49
51
53
55
57
59
61
63
65
67
69
71
73
75
77
Time Points

8.) Now let's create a simple shiny app which allows the user to select and correlate any time point verse another time point across all genes. To do this, we can create a server and ui function within the same file, paste both into the R session, then call them with:

>shinyApp(ui = ui, server = server)

Use the Iris dataset example from the lecture as a model. You can remove the kmeans clustering code and just focus on plotting the columns (time points) of the CDC15 data matrix against each other.

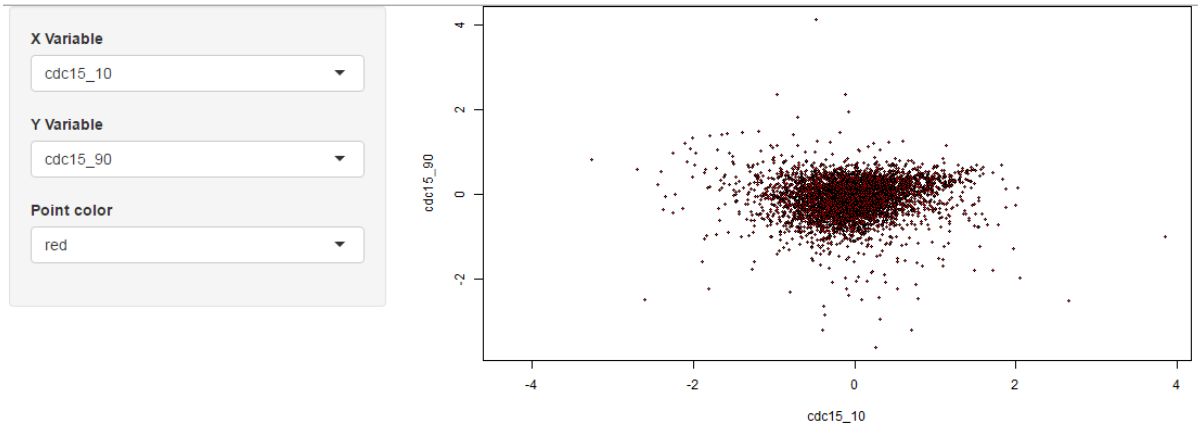
Hints:
The ui function can start with:

```
ui <-    fluidPage(
                  sidebarLayout(
                    sidebarPanel(
                          selectInput('xcol', 'X Variable', dimnames(dat)[[2]]),
```

The server function can start with:

```
server <- function(input, output) {
          selectedData <- reactive({
                dat[, c(input$xcol, input$ycol)]
          })
```

The web UI should look something like this:



```
> ui <-           fluidPage(
+     sidebarLayout(
+       sidebarPanel(
+         selectInput('xcol', 'X Variable', dimnames(cdc15)[[2]]),
+         selectInput('ycol', 'Y Variable', dimnames(cdc15)[[2]]),
+         selectInput('pcol', 'Point Color', colors)
+       ),
+       mainPanel(plotOutput("plot1")))
+     )

> server <- function(input, output) {
+     selectedData <- reactive({
+       cdc15[, c(input$xcol, input$ycol)]})
+     output$plot1 <- renderPlot({plot(selectedData(), col=c(input$pcol, "black"),
pch = 20)})}

> shinyApp(ui=ui, server=server)
```